VI-Zaštita i bezbednost podataka

- Računarska mreža predstavlja otvorenu javnu mrežu dostupnu svima.
- Uvek postoji mogućnost da neko neovlašćeno prati našu komunikaciju
- Mogućnost zloupotrebe je velika: krađa, promena, reklama, kontrola ...
- Veliki broj podataka koji se razmenjuje između računara ima svoju privatnost i potrebno je onemogućiti drugim osobama da im pristupe
- Bezbednost i zaštita podataka - jedan vrlo važan deo računarskih mreža
- Može se podeliti u nekoliko oblasti koje se međusobno prepliću:

- ***Poverljivost (tajnost)*** - sprečavanje otkrivanja sadržaja poruke
- ***Integritet informacija*** - sprečavanje neovlašćene izmene poruka
- ***Raspoloživost***-sprećavanje od neovlašćenog onemogućavanja pristupa
- ***Autentičnost informacija*** - definisanje i provera identiteta pošiljaoca
- ***Neporicanje*** - sprečavanje od lažnog poricanja slanja poruka

VI-Zaštita i bezbednost podataka

Sve ove probleme moguće je rešavati na različitim nivoima OSI sistema:

1. Na fizičkom nivou - prenosne linije mogu biti hermetički zatvorene u cevima sa argonom pod visokim pritiskom
2. Na sloju podataka - paketi mogu biti kodirani pri slanju i dekodirani po prijemu između dva čvora (ruter-ruter, host-ruter, ruter-host). Ovde su potencijalni problemi ruteri koji mogu postati meta napada.
3. Na mrežnom nivou - instaliranje firewall-ova(bezbedonosnih zidova) koji vrše filtriranje paketa
4. Na transportnom nivou - mogu se šifrirati poruke koje se razmenjuju između krajnjih tačaka (host-host): **SSL**-*Secure Sockets Layer*, **TLS** (*Transport Layer Security*)

- Nijedan od ovih slojeva ne rešava problem autentifikacije i neporicanja pa se rešenje mora potražiti na aplikativnom nivou: kriptografija i kriptografski algoritmi, substitucione tehnike, transpozicione tehnike, DES (*Data Encription Standard*), algoritmi za šifriranje sa javnim ključem, protokoli za autentifikaciju, digitalni potpisi, zaštitni zidovi

VI-Zaštita i bezbednost podataka

- **Kriptografija** – nauka o tajnom pisanju (šifrovanju), koja se bavi metodama očuvanja tajnosti informacija
- **Kriptografski algoritam** - postupak kojim se čitljiv tekst ***P*** transformiše u nečitljiv tekst ***C***
- **Kriptoanaliza** - nauka o dobijanju čitljivog teksta ***P*** (ili ključeva, ...)
- **Napad** - pokušaj kriptoanalize da bi se došlo do čitljivog teksta

Osnovni elementi od kojih se sastoji svaki kripto postupak:

1. **Tekst koji se šifruje** (*plaintext*)
2. **Algoritam šifrovanja** - postupak transformacije čitljivog teksta u oblik nečitljiv za onoga kome taj tekst nije namenjen
3. **Tajni ključ** - vrednost nezavisna od teksta koji se šifrira tako da isti algoritam generiše različit izlaz u zavisnosti od ključa koji se koristi
4. **Šifrovani tekst** (*ciphertext*)
5. **Algoritam dešifrovanja** - postupak dobijanja čitljivog teksta na osnovu šifriranog (kriptovanog) teksta

6.1-Kriterijumi podele kriptografskih sistema

1. U odnosu na **tip operacija** koje se koriste za šifriranje:

a) supstitucija - svaki element *plaintexta* (bit, slovo, grupa bitova ili slova) se preslikava (zamenjuje) u drugi elemenat

b) transpozicija - preuređenje elemenata u *plaintextu*

c) supstitucija i transpozicija

2. U odnosu na **broj ključeva** koji se koriste:

a) simetrični (konvencionalni) sistemi - obe strane(pošiljalac i primalac) koriste isti ključ: pošiljalac za šifriranje a primalac za dešifriranje.

b) asimetrični sistemi (ili sistemi sa 2 ključa, sistemi sa javnim ključem) ovde pošiljalac i primalac koriste različite ključeve, tj. ključ za šifriranje može biti javni, tj. dostupan svima, dok je ključ za dešifriranje tajni (privatni). Javni i privatni ključ čine par.

3. Način **procesiranja** (obrade) *plaintexta*:

a) šifriranje blokova - u jednom trenutku se ceo blok šifrira

b) šifriranje niza (*stream*) - šifrira se svaki element (bit, bajt ili reč)

VI-Zaštita i kontrola toka podataka

- U narednom razmatranju bavićemo se nekim aspektima koji utiču na bezbednost podataka u savremenim računarskim sistemima:

1. Algoritmi za šifrovanje i dešifrovanje podataka:

1. kripto sistemi sa simetričnim ključem,
2. kripto sistemi sa zaštičenim ključem
3. kripto sistemi sa javnim ključem.

2. Secure Socket Layer (SSL) i ***Transport Layer Security (TLS)***

- predstavljaju najčešće implementirane pristupe za osiguravanje transfera kod Interneta – *URL* ima oznaku *https*.

3. Računarski virusi i crvi

- mere za zaštitu od njih
- mrežne barijere (Firewall)

6.1-Algoritmi za šifrovanje podataka

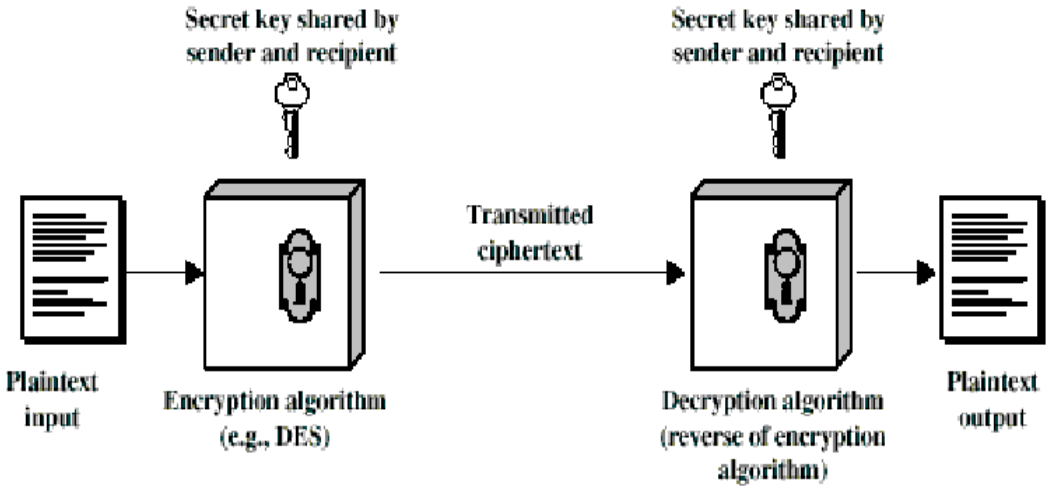
- Ako je **P** tekst koji treba šifrirati, **K** ključ za šifriranje, **E** algoritam za šifriranje, tada proces šifriranja možemo predstaviti kao:

$$KP = E_k(P)$$ **- gde je KP kriptovani podatak,**

a proces dešifrovanja kao:

$$P = D_{k1}(KP)$$ **- gde je D algoritam za dešifrovanje**

U opštem slučaju važi da je:



$P \quad E_k \quad KP \quad D_k \quad P$

6.1-Algoritmi za šifrovanje podataka

- Dobar algoritam za šifriranje treba da zadovolji sledeće kriterijume:
  1. cena razbijanja šifriranog teksta ne sme da prevazilazi vrednost šifrirane informacije
  2. da vreme potrebno za razbijanje šifre nije duže od vremena važenja informacije
- Vreme potrebno za razbijanje šifre (tj. pronalaženje ključa) zavisi od broja mogućih ključeva:

Veličina ključa u bitovima	***Broj mogućih ključeva(kombinacija)***	***Vreme potrebno za nalaženje ključa pri 1 enripciji/ μs***
32	$2^{32}=4.33x10^{9}$	$2^{31}\mu s=35.8$ min
56	$2^{56}=7.2x10^{16}$	$2^{55}\mu s=1142$ god
128	$2^{128}=3.4x10^{38}$	$2^{127}\mu s=5.4x10^{24}$ god

6.1-Algoritmi za šifrovanje podataka

Cezarovo šifrovanje(monoalfabetsko šifrovanje)

- predstavlja jedan od najranijih i najjednostavnijih načina za šifrovanje
- svaki karakter u poruci menja se drugim karakterom po nekom ključu
- svakom karakteru u poruci dodajemo na njegov ASCII kod neku vrednost ***n*** (kod originalnog Cezarovog šifrovanja dodaje se 3).
- Ako svakom slovu dodelimo numerički ekvivalent shodno poziciji u alfabetu, tj. A=0, B=1,…, Z=25, tada algoritam možemo opisati sledećom formulom: **C=E(P)=(P+3)mod26**
- Vrednost pomeraja može biti bilo koja veličina k ($1 \leq k \leq 25$), tako da opšti Cezarov algoritam možemo prikazati kao **C=E(P)=(P+k)mod26**,

Primer: *neka je plaintext koji treba šifrirati:* **VIDIMO SE U PETAK**

Šifrovani tekst glasi: **YLGLPR#VH#X#SHWDN**

Objašnjenje: **V**(ASCII kod je 56 h) ⟶ 56+3=59 a to je ASCII kod za **Y**

Ovo jednostavno kodiranje se malo koristi u ozbiljnim aplikacijama

6.1-Algoritmi za šifrovanje podataka

Polialfabetsko šifriranje

- Prevazilazi se loša strana monoalfabetskog šifriranja a to je učestalost pojavljivanja uobičajenih sekvenci.
- To je postignuto tako što se isti karakter ne menja uvek istim karakterom već je on promenljiv na osnovu nekog ključa.
- Zamenu možemo da izvršimo ne samo na osnovu konkretnog karaktera već i na osnovu njegove pozicije u poruci (**C(i)=P(i)+K+(i mod 3)** to znači da će karakterima na poziciji 0,3,6... da se doda 1, na pozicijama 1,4,7 ... dodaje se 2 a pozicijama 2,5,8 ... dodaje se 3.

Primer: **VIDIMO SE U PETAK** ⟶ **ZNJMRUZKZ&TJZEP**

Objašnjenje: **V** (56 h) ⟶ 56+3+1=5Ah a to je ASCII kod za **Z**

I (49 h) ⟶ 49+3+2=4Eh a to je ASCII kod za **N**

- Ovde se javlja problem ako je niz podataka koji se šifruju veliki jer se onda javljaju ponavljanja i određeni šabloni u kodiranju koje je lako uočiti a samim tim i dešifrovati.

6.1–Algoritmi za šifrovanje podataka

Transpozicione tehnike - Šifrovanje premeštanjem

- Svode se na neku vrstu permutacije slova u *plaintextu* bez njihovog menjanja
- Jedan od načina da se ovo uradi je da se karakteri iz poruke smeste u dvodimenzionalni niz sa *m* kolona. Prvih *m* karaktera smešta se u prvi red, drugih *m* karaktera u drugi i td. Zatim se izvrši permutacija kolona od 1 do *m* i tako se sada prenose ove kolone (na primer ako je *m=7* onda mogu da se prenesu kolone 4 3 1 2 5 6 7).

Varijanta 1: pisanje teksta u cik-cak obliku tako da se dobiju dve vrste. Šifrirani tekst se dobija čitanjem vrsta.

Primer: “napad odložen za dva sata popodne”

n p d d o e z d a a a o o n

a a o l ž n a v s t p p d e

šifrovana poruka glasi: **npddoezdaaaoonaaolžnavstppde**

6.1–Algoritmi za šifrovanje podataka

Varijanta 2:

- ✓ Upisivanje teksta u matricu vrsta po vrsta.
- ✓ Šifrirani tekst se dobija čitanjem kolona matrice pri čemu je moguće vršiti permutaciju kolona.
- ✓ Redosled kolona postaje ključ za algoritam šifriranja i dešifriranja.

	1	2	3	4	5	6	7
Ulaz:	N	A	P	A	D	O	D
	L	O	Ž	E	N	Z	A
	D	V	A	S	A	T	A
	P	O	P	O	D	N	E

Ako je ključ: **4 3 1 2 7 6 5** šifrovani tekst je:

AESO PŽAP NLDP AOVO DAAE OZTN DNAD

6.1-Algoritmi za šifrovanje podataka

Varijanta 3:

- Problem kod ovakvog šifrovanja je u tome što su zadržani originalni karakteri i to daje mogućnost da se ovaj kod lako razbije jer šifrirani tekst se lako prepoznaje jer ima istu frekvenciju pojavljivanja slova kao i izvorni tekst.
- Transpozicioni šifrator se može poboljšati tako što se transpozicija obavi više puta. Ako na prethodni tekst (**AESO PŽAP NLDP AOVO DAAE OZTN DNAD)** ponovo primenimo transpoziciju dobijamo:

	1	2	3	4	5	6	7
Ulaz:	A	E	S	O	P	Ž	A
	P	N	L	D	P	A	O
	V	O	D	A	A	E	O
	Z	T	N	D	N	A	D

Ako je ključ: **4 3 1 2 7 6 5** (može da se promeni) tada je šifrovani tekst:

ODAD SLDN APVZ ENOT AOOD ŽAEA PPAN

6.1-Algoritmi za šifrovanje podataka

Šifrovanje na nivou bita

- ovde se za šifrovanje koristi kodiranje na nivou bita.
- poruka se posmatra kao povorka bitova (1,0) i deli se na manje delove
- dužina sekvence je onoliko bitova koliko je dugačak ključ za šifrovanje
- nad svakom sekvencom vrši se operacija isključivo ili sa ključem za šifrovanje.
- na prijemnoj strani radi se ista operacija: kada se isključivo ili primeni dva puta nad istim podatkom dobija se isti podatak
- sigurnost ovakvog šifrovanja zavisi od dužine koda za šifrovanje
- ukoliko je dužina koda za šifrovanje veća veća je i sigurnost
- Moguće je taj kod menjati kod svakog slanja nove poruke

Ključ	**10011011**	***Ključ***	**10011011**
Podatak	**10101010**	***Šifrov.podatak***	**00110001**
Šifrovan podatak:	**00110001**	***Podatak***	**10101010**

Osnovni problem kod ovog šifrovanja što se zahteva da oba entiteta budu u stalnoj vezi kako bi mogli da razmenjuju šifre kodiranja.

6.2-Standardi za šifrovanje podataka

1. Data Encryption Standard (DES)

- Predstavlja primer blokovskog šifrovanja.
- Poruka se deli na 64 bitne blokove (56+8 bita za kontrolu parnosti).
- Koristi se 56-bitni ključ i složene operacije premeštanja bitova.
- Postoji 19 koraka kroz koje prolaze podaci koji se šifruju.
- Napravljen je *DES Cracker* koji vrši razbijanje koda $2^{56}=7{,}2 \times 10^{16}$
- Novi trostruki *DES* koji je koristio gotovo isti način kodiranja
- Korišćen 168-bitni ključ (3x56 bitova).
- Tri puta sporije od normalnog *DES-a*.

2. Advansed Encryption Standard (AES) i Rijndael algoritam

- Ovde se koriste ključevi za kodiranje i to od 128, 192 i 256 bita.
- Za ključ od 256b postoje $1{,}1^{x}10^{77}$ mogućih ključeva - ogromna količina
- Ima jačinu trostrukog *DES*-a ali se ovde operacije izvode mnogo brže.

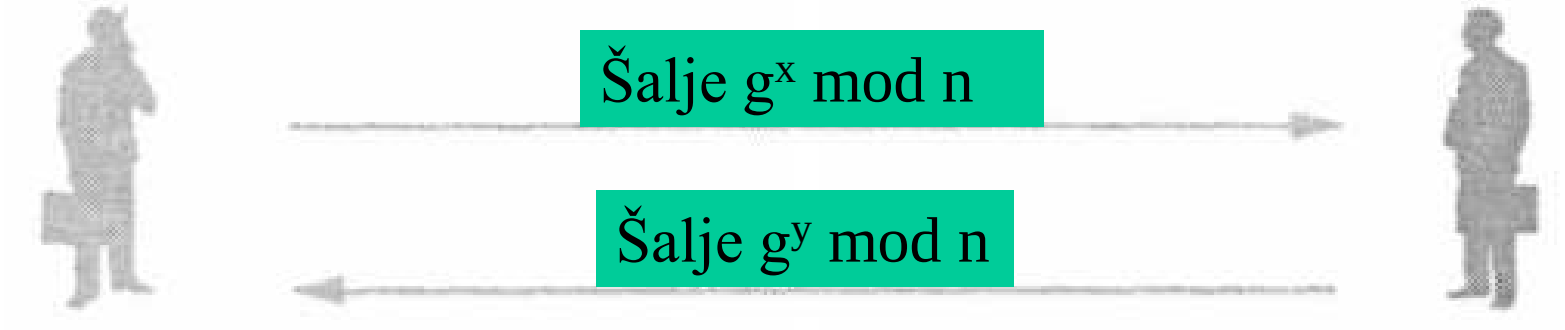
3. Clipper Chip i Skipjack algoritam

- Predstavlja specijalno dizajniran čip koji je mogao da se koristi u bezbednosnim komunikacijama koje su išle telefonskom vezom.

6.2-Distribucija i zaštita ključa

Kako osigurati diskretnost ključa kod njegove distribucije do odredišta?

1. ***Shamirov* metod** – ključ se deli na više osoba tako da je potrebno da se oni skupe zajedno da bi se dobio ključ za šifrovanje.
2. ***Diffie-Hellman-ova* razmena ključa** – ovde pošiljalac i primalac razmenjuju izračunate vrednosti na osnovu kojih se može utvrditi vrednost ključa za šifrovanje.



Treća strana zna g, n, g^xmod n ili g^ymod n,
ali, i pored toga, ne može lako da utvrdi **K**.

A

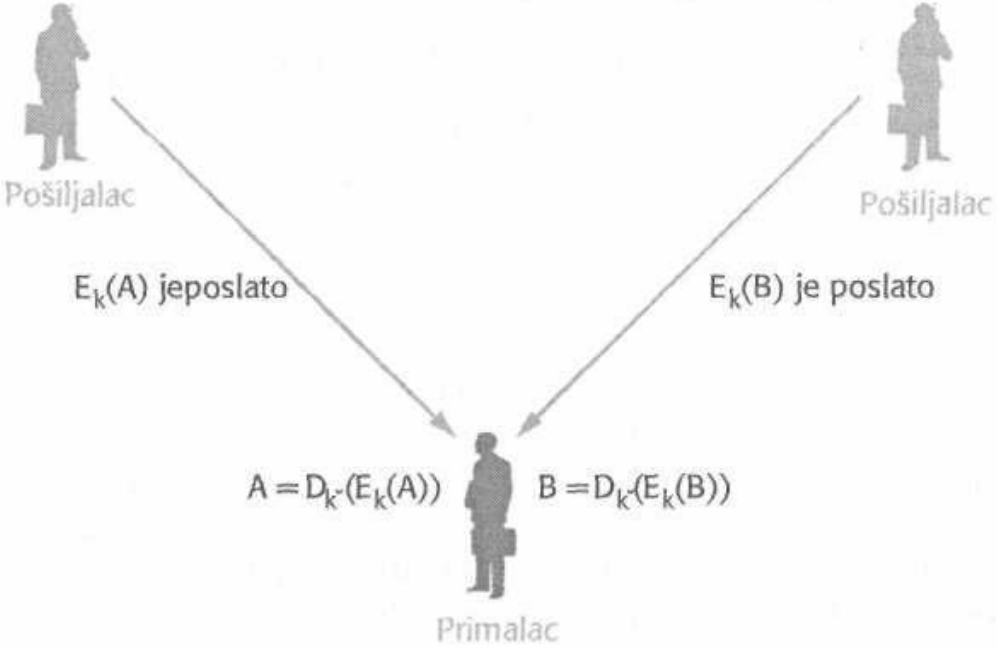
Bira **x** i čuva ga u tajnosti.
Izračunava **$K=g^{xy}$ mod n**.

B

Bira **y** i čuva ga u tajnosti.
Izračunava **$K= g^{xy}$ mod n**.

6.3 – Šifrovanje javnim ključem

- Predstavlja metod da je algoritam i ključ šifrovanja dostupan svakome
- Koristi se u bankama za finansijsko poslovanje klijenata kao i na WEB-u
- Najtipičniji predstavnici ovakvog načina šifrovanja podataka su: ***RSA*** algoritam koji je zasnovan na matematičkoj teoriji brojeva, digitalni potpisi koji predstavljaju jedan od načina autentifikacije klijenta.
- Ovde se poruka šifruje tako da samo pošiljalac zna način na koji je to urađeno i samo on može da izvrši dešifrovanje poruke.

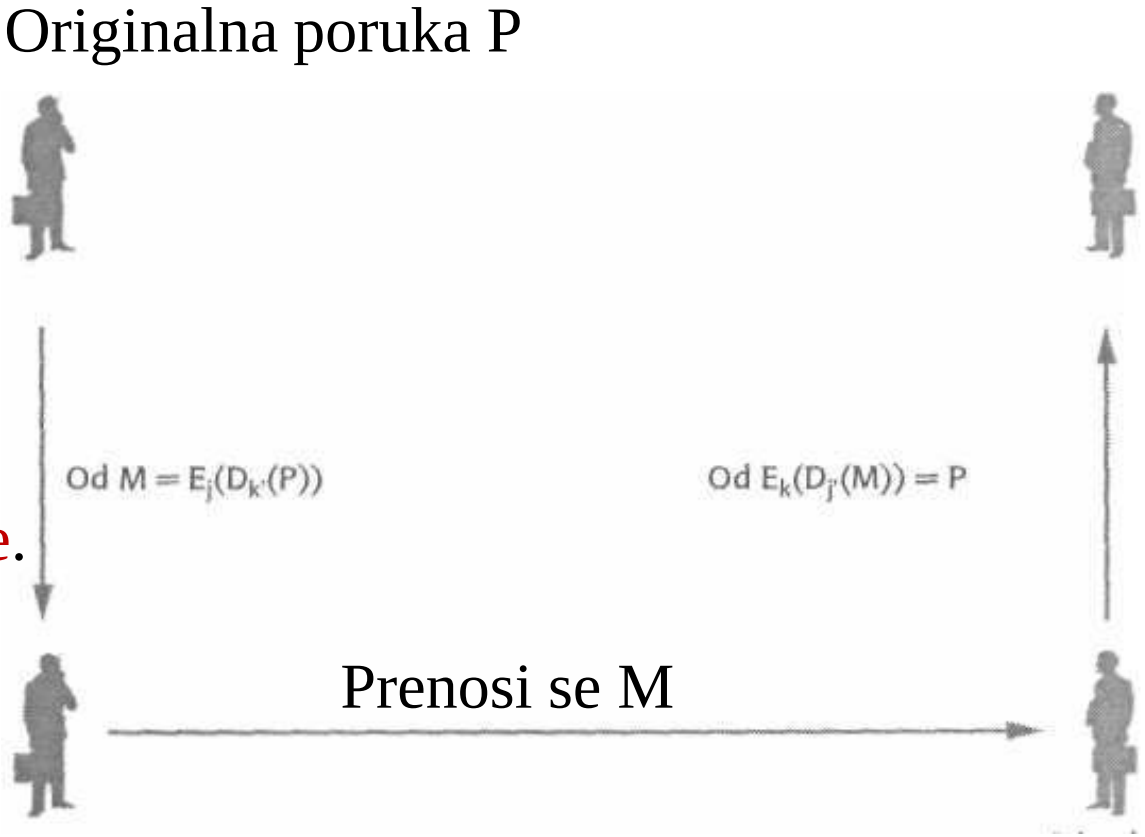


6.3 – Digitalni potpis

- Šifrovanje poruke tako da način šifrovanja zna samo pošiljalac
- Metod koristi dva para metoda šifrovanja javnim ključem/dešifrovanja koja označavamo kao ($\mathbf{E_k}$, $\mathbf{D_{k'}}$) i ($\mathbf{E_j}$, $\mathbf{D_{j'}}$) gde su **j** i **k** javni ključevi, a **k'** (zna samo pošiljalac) i **j'** (zna samo primalac) privatni ključevi

Svrha digitalnog potpisa:

- da potvrdi autentičnost sadržaja poruke (dokaz da poruka nije promenjena na putu od pošiljaoca do primaoca),
- da obezbedi garantovanje identiteta pošiljaoca poruke.
- Algoritam koji je postao standard za digitalni potpis je **RSA** algoritam.



6.3 – Digitalni potpis

- Funkcija sažetka **H** (*hash*) je jednosmerna funkcija koja ulazni niz proizvoljne dužine, poruku m, pretvara u niz fiksne dužine (najčešće 128-256 bita) koji se naziva sažetak poruke **h** (*message digest*).

$$\mathbf{h=H(m)}$$

- Iz dobijenog sažetka poruke **h** je nemoguće dobiti izvornu poruku m, zbog jednosmernosti funkcije sažetka, te je na taj način osiguran **integritet** poruke.
- Svojstvo funkcije sažetka je da i najmanja promena orginalne poruke uzrokuje drastične promena u njenom sažetku.
- Verovatnoća da se za dve različite poruke generiše isti sažetak je jako mala.
- Sažetak poruke je i prvi korak u kreiranju digitalnog potpisa - izračuna se sažetak orginalnog dokumenta koji se digitalno potpisuje.
- Najčešće korišćene funkcije sažetka su **SHA** (*Secure Hash Algorithm*) i **MD5** (*Message Digest*).

6.4 – Zaštita na transportnom sloju

Koriste se dva protokola i to:

1. ***Secure Socket Layer*** (SSL)
2. ***Transport Layer Security*** (TLS)

- Oba protokola se nalaze između sloja aplikacija i transportnog sloja i osnovna uloga im je:
  1. da obezbede uslove za bezbednu razmenu informacija putem šifrovanja istih,
  2. da obezbede autentičnost servera kako bi korisnici sigurno znali da je to pravi server - **sertifikat X.509**.
- Sertifikat predstavlja dokumet koji obezbeđuje sajt na koji smo se prijavili a on tačno definiše podatke o sajtu:
  - ✓ko je izdao sertifikat,
  - ✓vremenski rok važenja sertifikata
  - ✓dva otiska koji predstavljaju digitalni potpis koji nam služi za autentifikaciju sajta.

6.4 – Zaštita na transportnom sloju

- SSL obezbeđuje *privatnost, integritet podataka i autentičnost pošiljalaca* korišćenjem kombinacije šifrovanja javnim ključem, simetričnog šifrovanja, digitalnih sertifikata
- Transakcija korišćenjem SSL protokola uključuje sledeće aktivnosti:
  1. server šalje svoj digitalni sertifikat klijentu
  2. klijent proverava da li je sertifikat izdat od strane CA
  3. klijent i server razmenjuju javne ključeve
  4. klijent generiše tajni ključ koji se koristi samo u započetoj transakciji
  5. klijent šifruje generisani tajni ključ, korišćenjem serverovog javnog ključa i šalje ga serveru
- U daljem toku transakcije server i klijent koriste isti tajni ključ metodom simetričnog šifriranja (šifriranje sa zajedničkim tajnim ključem, npr. DES algoritam)
- U verziji 2.0 SSL podržava samo proveru autentičnosti servera, dok je u novoj SSL v3.0 uključena i podrška za proveru klijenta.

6.5 – Firewall – Zaštitini zid

Firewall – služi nam da odvoji lokalnu mrežu od spoljneg uticaja.

- Predstavlja poseban računar koji jednostavno kontroliše celokupni saobraćaj i vrši se eliminisanje sumnjivih paketa kako bi se zaštitila privatnost lokalnih mreža.
- Funkcioniše na principu kontrole pojedinih IP adresa ili adresa portova

Firewalli primarno funkcionišu koristeći tri osnovna metoda:

1. **Filtriranje paketa** - odbacuje TCP/IP pakete neautentifikovanih hostova kao i pri pokušaju povezivanja na neautentifikovane servise.
2. ***Network Address Translation*** (NAT) - Prevodi IP adrese unutrašnjih hostova i tako ih skriva od spoljnog praćenja. Ovaj metod se naziva i maskiranje IP adrese (*IP address masquerading*).
3. **Proxy servisi** - uspostavljaju konekcije na visokim aplikacionim nivoima za unutrašnje domaćine u cilju da se kompletno prekine konekcija mrežnog sloja između unutrašnjih i spoljnih domaćina.

6.6 – Pretnje i napadi

➢ ***Denial of service*** (DoS) napad odbijanja servisa predstavlja danas jedan od najčešćih napada - *smurf napad* i *napad poplave SYN segmenata*

1. Smurf napad - bazira se na mrežnoj naredbi *ping*.

✓Ovde se koristi ***Echo Request*** (odgovor prozvanog računara) koji je prepravljen tako što je kao odredišna adresa stavljena emisiona adresa, tako da se paket šalje do svih adresa u okviru mreže, a u polje izvorne adrese stavlja se adresa napadnutog računara.

✓ Sada svaka mašina koja dobije *Echo* zahtev reaguje tako što vraća *Echo Replay* do napadnutog računara.

2. Zasniva se na protokolu koji TCP koristi za postavljanje konekcije.

✓Ovde klijent šalje TCP SYN segment kojim zahteva konekciju.

✓ Server šalje potvrdu da je prihvatio zahtev i nakon toga klijent nastavlja konekciju

✓ Međutim, ako se pošalje zahtev sa lažnom IP adresom, i ne čeka se odgovor od servera već se odmah uputi novi zahtev može se desiti da se napadnuti server blokira.

7.1 – Kontrola toka podataka

- Kako preneti veoma duge poruke ?
- Na koliko okvira treba podeliti neku poruku
- Kolika je dužina okvira optimalna?
- Kako reagovati na oštećene prenose?
- Da li preneti celu ili deo poruke?
- Kako primalac da obavesti pošiljaoca koji deo poruke je loš i šta raditi ako se desi da ta poruka ne stigne do pošiljaoca?
- Kako rešiti problem ako računari pošiljaoca i primaoca ne funkcionišu na istoj brzini ?
- Šta se dešava ako se poslati okvir izgubi u toku prenosa a primalac to ne zna da je izgubljen ?
- Šta se dešava ako i primalac i pošiljalac istovremeno šalju okvire ?

O svim ovim pitanjima sloj veze ili sloj podataka, u saglasnosti sa OSI referentnim modelom, treba da da odgovore.

7.1 – Kontrola toka podataka

- Sve ove zadatke možemo da grupišemo u 4 celine koje će predstavljati osnovne funkcije koje treba da ostvari nivo veze ili nivo podataka.

Ove funkcije odnose se na:

1. obezbeđivanje servisa za potrebe mrežnog nivoa,
2. formiranje okvira tj. određivanje načina grupisanja bitova fizičkog nivoa u okvire,
3. kontrola grešaka u prenosu,
4. kontrola toka podataka koja se odnosi na regulisanje toka prenosa okvira tako da se i spori prijemnici ne "prenatrpaju" porukama od strane brzih predajnika.

7.1.1 Servisi na mrežnom nivou

(1) **nepotvrđivanje okvira** - izvorna stanica predaje nezavisne okvire odredišnoj pri čemu ih odredišna ne potvrđuje. Na početku i na kraju prenosa ne uspostavlja se i ne raskida veza, respektivno. Ako se signal izgubi zbog šuma na liniji, ne zahteva se novi pokušaj za slanje. Ovaj tip prenosa koristi se kod pouzdanih komunikacija.

(2) **potvrđivanje okvira** - svaki okvir se predaje i potvrđuje individualno. Ovakvim načinom rada predajnik zna da li je okvir korektno ili ne stigao do prijemnika. Ako za specificirani vremenski period okvir nije stigao do prijemnika on se šalje ponovo. Ovaj tip usluga pogodan je za korišćenje kod nepouzdanih veza kakve standardno srećemo kod bežičnog prenosa.

(3) **potvrđivanje brojivih poruka** - izvorišna i odredišna stanica pre početka prenosa uspostavljaju najpre vezu. Zatim, u toku prenosa, svaki predati okvir se numeriše. Usluga na nivou veze garantuje da će se svaki okvir korektno primiti i da će okviri biti primljeni u ispravnom redosledu. Kod ovog tipa prenosa identifikuju se tri faze.

7.1.2 – Formiranje okvira

1. **brojanje znakova** - određenim poljem u okviru zaglavlja specificira se broj karaktera u okviru. Na donjoj slici, radi pojednostavljenja, zaglavlje čini samo znak koji se odnosi na broj karaktera u okviru.

Ispravno formirani okviri

6 4 3 2 1 6 3	**9** 3 7 4 8 9 1 0 5 6	**4** 0 1 7 8	**8** 0 2 4 6 1 3 5 7
6 karaktera	9 karaktera	4 karaktera	8 karaktera

Neispravno formirani okviri

6 4 3 2 1 6 3	**8** 3 7 4 8 9 1 0 5	**6** 4 0 1 7 8 8	**0** 2 4 6 1 3 5 7
okvir 1	okvir 2	novi karakter-broj	

- Kada nivo-veze na odredišnom kraju izdvoji iz zaglavlja broj-karaktera, on zna koliko karaktera slede.
- Problem kod ovog algoritma se javlja ako dođe do greške u prijemu broja karaktera jer prijemnik izlazi iz sinhronizacije i neće biti u stanju da locira početak novog okvira.
- Iz ovog razloga ovaj metod se retko koristi.

7.1.2 – Formiranje okvira

2. **karakteri početak i kraj prenosa, sa umetanjem karaktera**

- početak prenosa svakog okvira karakteriše ASCII karakter sekvenca **STX DLE**, a kraj sekvenca **DLE ETX**, (DLE je *Data Link Ecape*, STX je *Start of Text*, a ETX je *End of Text*).
- Ozbiljan problem kod ovog metoda se javlja kod prenosa binarnih podataka jer može da se dogodi da se par karaktera STX DLE ili DLE ETX javi u okviru podataka što ima za efekat narušavanje granica okvira
- Rešava se tako što predajnik na nivou veze (*link layer*) pre svakog ASCII DLE karaktera umeće (ubacuje) u niz karaktera po još jedan DLE karakter – tehnika **umetanje-karaktera** (*character stuffing*).

Primer:

podaci koji se predaju: A DLE B

podaci nakon umetanja: STX DLE A **DLE** DLE B DLE ETX

podaci nakon prijema: A DLE B

7.1.2 – Formiranje okvira

3. **umetanje bita**

- Okvir podataka može da sadrži proizvoljan broj bitova, a takođe i dozvoljen je i proizvoljan broj bitova po svakom okviru.
- Svaki okvir počinje i završava specijalnim bit oblikom **01111110** - ***flag*-bajt**.
- Kada predajnik na nivou-veze u informacionom delu naiđe na pet uzastopnih jedinica on ubaci 0 u izlaznom nizu bitova.
- Kada prijemnik naiđe na pet uzastopnih jedinica, iza čega sledi bit 0, on automatski izbacuje umetnuti 0 bit.
- tehnika koju nazivamo **umetanje bita** (*bit stuffing*) gde su umetnuti bitovi podvučeni.

Umetnuti bitovi

Primer:

početni podaci: 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

na liniji: **01111110** 0 1 1 0 1 1 1 1 1 **0** 1 1 1 1 1 **0** 1 1 0 0 1 0 **01111110**

podaci nakon obnavljanja: 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

7.1.3 – Kontrola greške

- Fizički nivo prihvata serijski niz podataka i prosleđuje ih odredištu.
- Za niz podataka niko ne garantuje da je ispravan tj. da nema grešaka.
- Broj koji je primljen može biti manji, jednak ili veći od predatih bitova
- Kontola grešaka definiše kako uređaj proverava da li u okviru postoje greške i šta se radi ako se one otkriju.
- Najčešće korišćena tehnika za kontrolu grešaka je bazirana na dve f-je:

1. detekciji grešaka (kontrola pariteta, ček suma ili ciklični kod-CRC),
2. zahtevu za automatskim ponavljanjem - ARQ (*Automatic Repeat Request*)–automatski zahtev za ponovnim slanjem.

Tu razlikujemo tri različita metoda ARQ:

1. pozitivna potvrda (*acknowledgment*, ACK) – prijemnik vraća pozitivnu potvrdu o tome da je okvir stigao bez greške,
2. ponovno slanje nakon isteka određenog vremena (*timeout*) – predajnik ponovo šalje okvir ako je isteklo unapred dogovoreno određeno vreme,
3. negativna potvrda i ponovno slanje (*not acknowledgment*, NACK) – prijemnik vraća negativnu potvrdu za okvire koje nisu došli ili su stigli oštećeni tako da predajnik opet šalje te iste okvire.

7.1.4 – Kontrola toka podataka

Najjednostavniji način kontrole toka podataka radi na sledeći način:

1. primalac pokazuje „spremnost" da prihvati podatke slanjem upita
2. pošiljalac šalje podatke.
3. po prijemu primalac mora da pokaže spremnost da primi nove podatke.

➢ Ova procedura je pogodna kada se poruka šalje kao jedna celina

➢ Uobičajeno je da pošiljalac deli veliku poruku u manje delove (celine)

➢ Ovo se radi iz sledećih razloga:

✓ što je duži prenos veća je verovatnoća da će doći do greške, što dovodi do potrebe za ponovnim slanjem (retransmisijom) te celine.

✓ Kada su celine manje, manja je verovatnoća greške a i kod njene pojave, količina podataka koje treba ponovo poslati je manja.

✓ na liniji sa više tačaka poželjno je ne dozvoliti stanici da zauzima liniju dugo i time izazove velika kašnjenja drugim stanicama,

✓ veličina memorijskog bafera prijemnika može da bude ograničena.

7.2 Signaliziranje

- Ovde pošiljalac šalje podatke stalno sve dok ga primalac ne obavesti da prestane jer nije iz nekog razloga u mogućnosti da prihvata podatke.

1. **DTE-DCE** – kada DTE želi da pošalje podatke šalje se prvo signal **RTS** kojim se traži dozvola da se podaci šalju. Ako je sve u redu DCE odgovara sa **CTS** kojim se dozvoljava slanje podataka.
2. **X-ON/X-OFF** - ovde se podatak o zahtevu za slanjem i spremnost za prijem šalju u okviru podataka kao deo protokola – *in band signaling* ili signaliziranje u opsegu.

- ASCII set karaktera definiše dva kontrolna karaktera za kontrolu toka i to DC3 (13h) i DC1 (11h) koji se nazivaju još i X-OFF i X-ON respektivno (CTRL-S i CTRL-Q sa tastature).
- Uobičajena upotreba ovog protokola je kod listanja velikih fajlova na ekranu.
- Tada možemo putem CTRL-S da stopiramo prikazivanje na ekranu tj. da ga zaustavimo a sa CTRL-Q da nastavimo njegovo prikazivanje.

7.3 – Kontrola orijentisana okvirima

- Kod sinhrone komunikacije potrebno je nešto veće organizovanje jer se prenos vrši u grupama-okvirima i prenosi se bitovi a ne bajtovi
- Većina protokola deli informacije na manje okvire koji se onda šalju.
- Znači ovde se javlja tkz. dvostepeno ili dvoslojno slanje podataka.
- Korisnik sakuplja podatke (sloj mreže) i predaje ih pošiljaocu (sloj podataka)
- Na drugoj strani primalac prima te podatke (sloj podataka) i dostavlja ih korisniku kome su ti podaci i poslati (sloj mreže).
- Kontrola toka ovde postoji na različitim slojevima

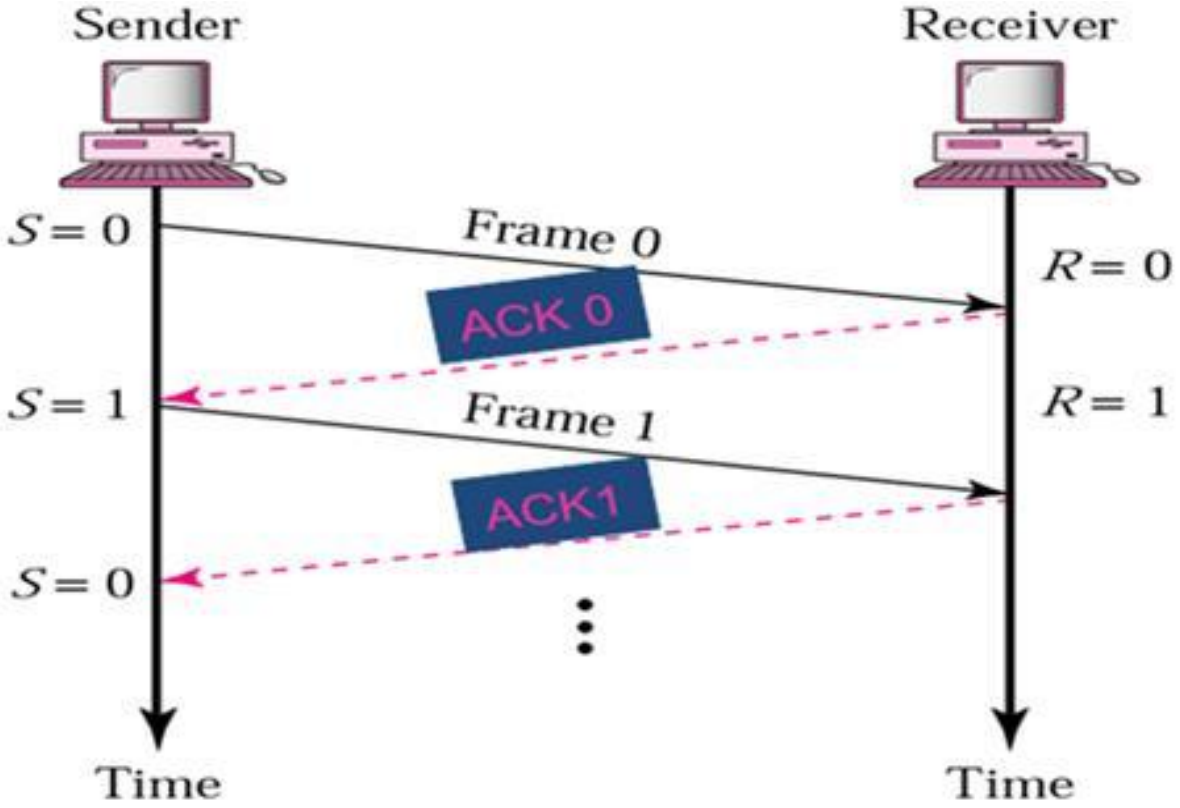
1.Neograničeni protokol

- Ovde se pretpostavlja da primalac ima neograničeni bafer i da je njegov kapacitet dovoljan da prihvati celokupnu poruku koja je upućena.
- Primalac stalno vrti petlju u kojoj ispituje da li ima podataka za prijem.
- Ne razmatra se da li su podaci stigli do odredišta i kako su stigli jer je predpostavka je da su svi paketi ispravno stigli i istim redosledom.

7.3.2 – Protokol stop and wait

2. Protokol *stop-and-wait*

- Svaki put kada primalac primi paket on šalje potvrdu nazad do pošiljaoca. Poruka predstavlja takođe poseban okvir koji se šalje.
- Pošiljalac sve dok ne dobije tu potvrdu ne šalje drugi okvir.
- Za razliku od neograničeng protokola koji šalje maksimalni broj okvira u jedinici vremena, ovaj protokol šalje minimalan broj okvira.



7.3.2 – Protokol stop and wait

Nedostaci protokola *stop-and-wait*

- ✓ Ako se poslati okvir izgubi primalac nikada ne šalje potvrdu i pošiljalac čeka beskonačno dugo.
- ✓ Ako se izgubi potvrda primaoca, dešava se isto.
- ✓ Ako se potvrda ošteti, pošiljalac može da donese pogrešan zaključak i tako dolazi do greške.
- ✓ Pošiljalac neće sigurno zasuti primaoca sa ogromnim brojem okvira ali zato može doći u suprotnu krajnost: da provede veliki deo u čekanju
- ✓ Kada se koristi više celina (ramova) za jednu poruku jednostavna procedura „stop and wait" ne obezbeđuje dobru iskorišćenost linija.

7.3.3 – Merenje efikasnosti protokola

- Pojam **efikasnost protokola** koja se može meriti na nekoliko načina.
- Najvažniji su veličina bafera za smeštanje primljenih podataka kao i brzina kojom se oni primaju.
- Pojam **efektivne brzine prenosa podataka** predstavlja stvarni broj bitova podataka koji se pošalje u jedinici vremena i ona iznosi kod:
- ***neograničeni protokol***: **T+F/R**
- ***stop-and-wait***: **(T+F/R+D/S) + (T+A/R+D/S) = 2(T+D/S)+(F+A)/R**
- gde je: **R**-bitska brzina, **S**-brzina signala, **D**-rastojanje između pošiljaoca i primaoca, **T**-vreme potrebno za kreiranje jednog okvira, **F**-broj bitova u okviru, **N**-broj bitova podataka u okviru, **A**-broj bitova u potvrdi.
- Vidi se da je efektivna brzina prenosa podataka zavisna od mnogih parametara kao što su vrsta protokola, veličina okvira, rastojanje koje se prelazi i td.

7.4 – Protokoli klizajućih prozora

- Svaki okvir se posebno numeriše i definiše se veličina prozora po broju okvira koji se šalju (posebno za pošiljaoca i posebno za primaoca)
- Ako definišemo da se svaki prozor sastoji od ***I*** okvira počevši od okvira W (W-I), tu važe sledeća pravila za neki prozor:

1. Okvir koji je numerisan brojem manjim od **W** je već poslat i potvrđen.
2. Ni jedan okvir koji je numerisan brojem većim od **W+I** još nije poslat.
3. Ako su svi okviri iz prozora poslati, za okvire za koje nisu stigle potvrde za njih kažemo da su nerešeni okviri (*outstanding frames*).
4. Inicijalno prozor sadrži okvire koji startuju sa okvirom 0. Kako korisnik obezbeđuje nove pakete tako se prozor proširuje.
5. Ako se ne dobije potvrda za neki okvir on ostaje u prozoru. Kada se potvrda dobije tada se okvir izbacuje iz prozora ali samo pod uslovom da ispred njega ne postoji neki okvir koji nije dobio potvrdu.
6. Max. veličina prozora definiše i broj okvira koji mogu biti nerešeni.
7. Ako je prozor veličine 1 imamo *stop and wait* protokol, a ako je prozor veći od ukupne količine okvira za slanje imamo neograničen protokol.

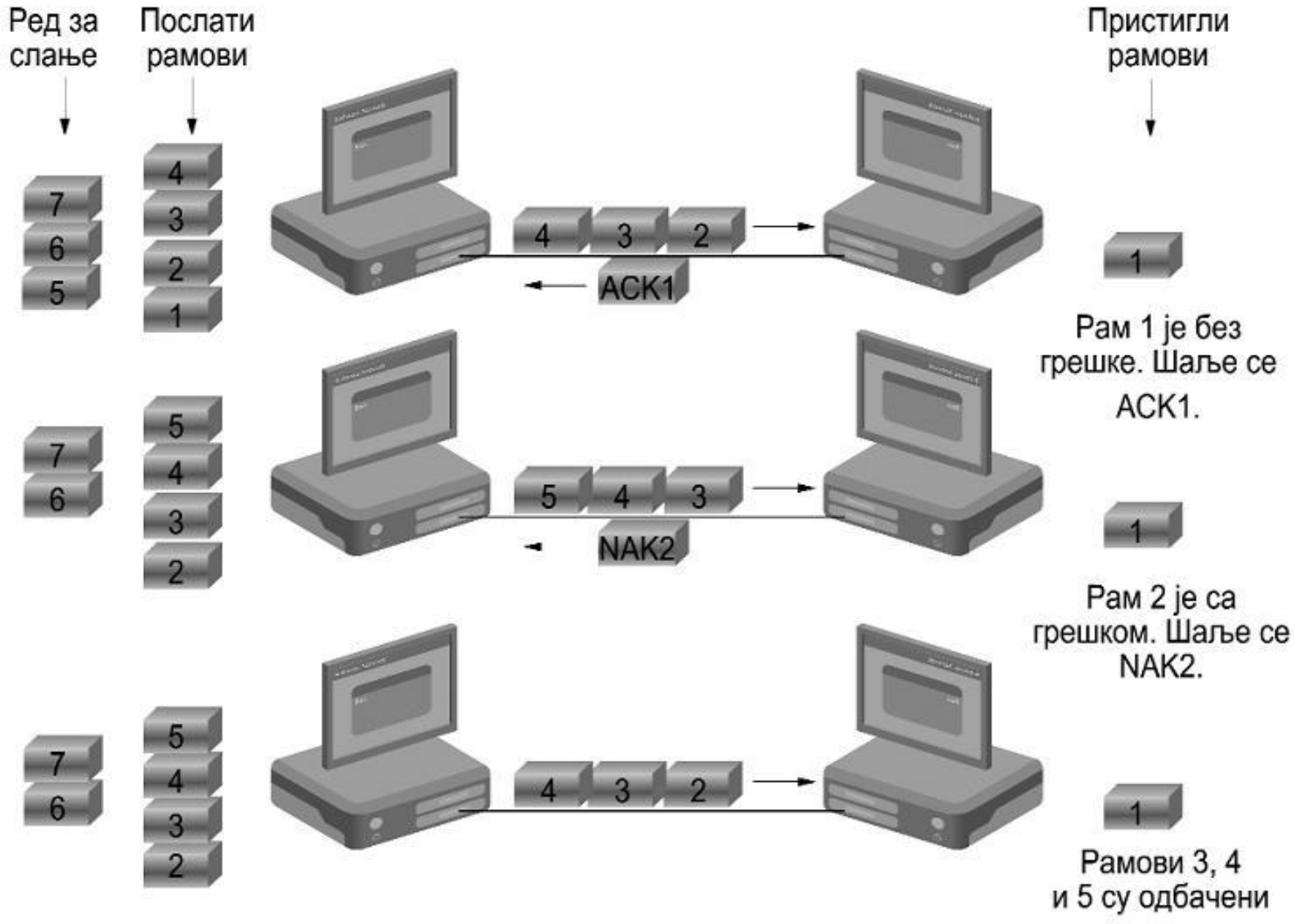
7.4 – Protokoli klizajućih prozora

- Postoje dve vrste ovog protokola i to **Go Back n** koji zahteva prijem okvira istim redosledom kojim su i poslati, i **protokol selektivne retransmisije** koji to ne zahteva.
- Potrebno je definisati format okvira koji se šalje koji ima sledeća polja:
  - Izvorna adresa – predstavlja adresu uređaja koji šalje okvir
  - Odredišna adresa – adresa na koju se šalje okvir
  - Broj okvira – svaki okvir ima sekvencu brojeva koji počinju od 0. Ako ovo polje ima K bitova, znači da ima mogućnost za ukupno 2^K-1 okvira.
  - ACK– celobrojna vrednost koja predstavlja broj okvira koji je potvrđen. Nije potrebno da se šalje poseban okvir već je moguče da se ta poruka pošalje sa podacima – ***piggybacking***.
  - Tip okvira – predstavlja vrstu okvira koji se šalje (ACK, NAK).
  - Podaci – korisna poruka.
  - CRC – provera grešaka u poslatoj poruci.

7.4.1 – Go Back n protokol

- Brojevi okvira moraju da se nalaze između **0 - 2^K-1** gde je K broj bitova u polju broj okvira.
- Ako ima više okvira onda dolazi do dupliranja brojeva
- Prijemni uređaj prihvata okvire u skladu sa redosledom brojeva
- Ako je pristigli okvir oštećen prijemni uređaj ga ignoriše i šalje NAC.
- Prijemni uređaj ne šalje potvrdu za svaki okvir već to može da uradi za svaki drugi, treći ...
- Okviri primljeni između ovih potvrda se smatraju kao ispravno primljeni okviri.
- Uređaj koristi ***piggyback*** pristup svaki put kada je to moguće.
- Uređaj pošiljaoca baferuje pakete iz svih okvira u prozoru u slučaju da ih treba ponovo da šalje.
- Ako uređaj ne primi potvrdu u određenom periodu predpostavlja se da je nešto loše bilo na liniji.
- Ovde se postavlja tajmer okvira koji ima ulogu odbrojavanja unazad.

7.4.1 – Go Back n protokol



7.4.1 – Go Back n protokol

Primer: *Pretpostavimo da **A** šalje 8 okvira numerisanih (0-7) uređaju **B**.*

- Ako je veličina prozora 2^K (K=3) može da dođe do greške u protokolu.
- Uređaj B primi tih osam okvira i pošalje potvrdu da je primio (ACK7).
- Zbog problema na vezi uređaj A ne primi zadnju potvrdu (ACK7).
- Posle određenog *timeout*-a uređaj A ponovo šalje okvire 0-7 jer nije dobio potvrdu za prethodno slanje.
- Sada uređaj B očekuje okvir 0 od novog prozora a prima 0 od starog prozora koji mu A ponavlja.
- Očigledno da se ovde javlja problem.
- Redukovanjem broja okvira u prozoru za 1 (2^K-1) ovaj problem može da se izbegne (sada se šalju okviri od 0-6).
- Kako sada uređaj B očekuje nove okvire 7, 0-5 a dobija okvire 0-6 jer A nije primio ACK6, ponavlja iste okvire, zna da je došlo do greške
- Ovde sada nema dupliranja brojeva okvira u prozoru jer svaki okvir u prozoru ima svoj jedinstven broj.

7.4.2 – Selektivna retransmisija

Sličnosti sa prethodnim protokolom:

- ✓ Formati okvira su slični i okviri se numerišu pomoću K-bitnog polja.
- ✓ Pošiljalac ima prozor definisan maksimalnim brojem nerešenih okvira.
- ✓ Protokol selektivne retransmisije uvek šalje potvrdu zajedno sa novim podacima, i ako je moguće ne izdaje eksplicitne potvrde za sve okvire.
- ✓ Ako je okvir potvrđen uređaj pošiljaoca smatra da su primljeni i svi okviri pre njega.
- ✓ Protokol koristi NAK okvire za sve oštećene okvire, ili okvire koji su stigli van redosleda.
- ✓ Koristi tajmere za slanje specijalnih potvrda za okvire u periodama slabog saobraćaja i za ponovno slanje okvira koji nisu potvrđeni duže vremena.

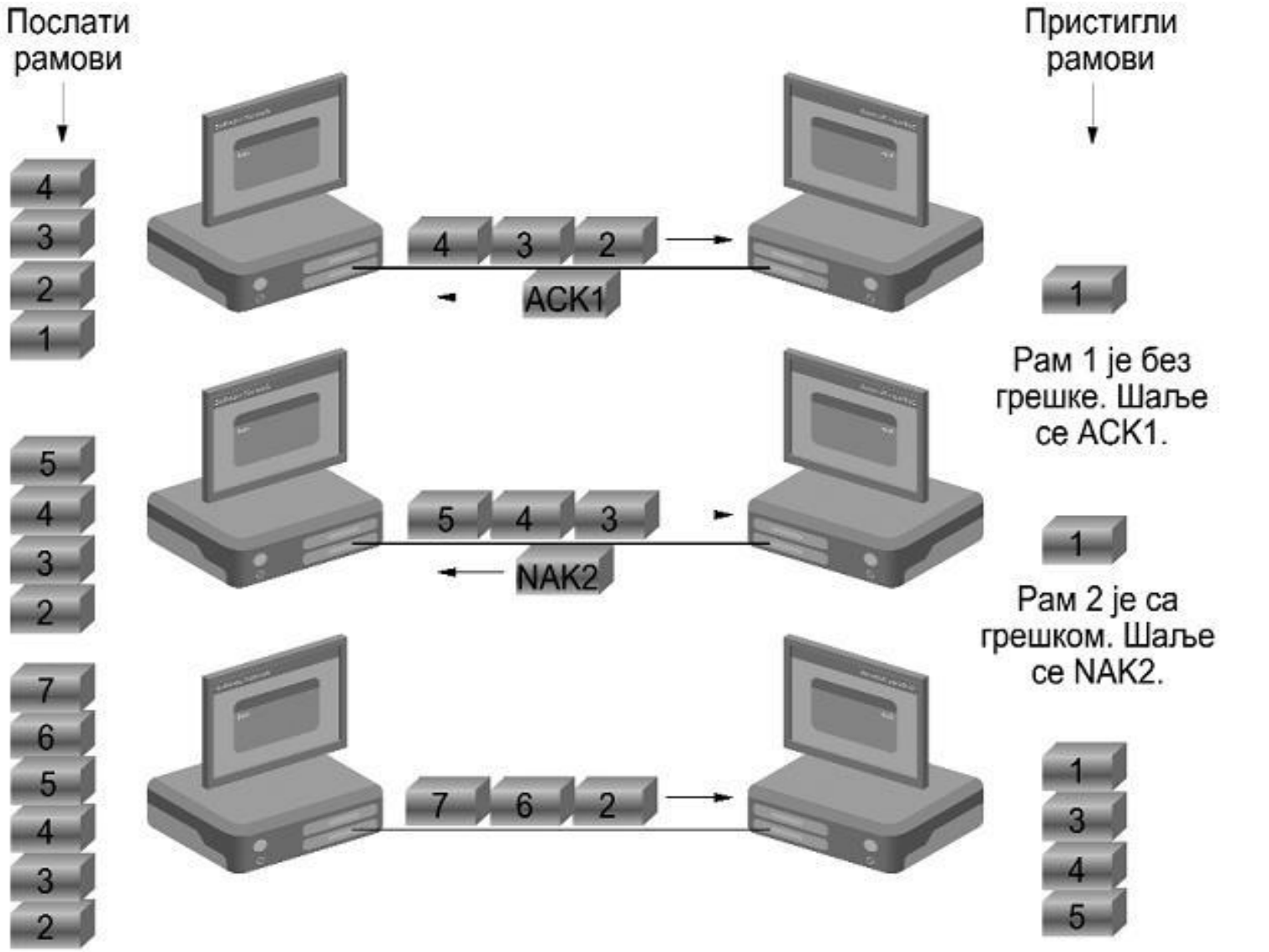
Najveća razlika je u tome što ovde postoje dva prozora, i to po jedan za predaju i jedan za prijemnu stranu protokola.

7.4.2 – Selektivna retransmisija

Razlike:

- ✓ Ako se pristigli okvir nalazi u prijemnom prozoru, on se baferuje ali se on ne daje se korisniku sve dok ne stignu svi njegovi prethodnici.
- ✓ Svaki put kada stigne okvir van redosleda, protokol šalje NAK za okvir koji je očekivao.
- ✓ Razlog za to je činjenica da je došlo do promene redosleda u primanju okvira, da se nešto desilo sa očekivanim okvirom.
- ✓ Ako istekne vreme za potvrdu okvira ponovo se šalju samo okviri čiji su tajmeri istekli.
- ✓ Ako protokol primi NAK ponovo šalje naznačeni okvir.
- ✓ *Piggyback* potvrda ne mora da se odnosi na najskorije primljeni okvir.
- ✓ Potvrda može da se odnosi i na okvir koji je ranije primljen ali je po redosledu iza ovoga koji je primljen.
- ✓ Ovde postoji ograničenje veličine prozora i to ako je max. veličina prozora na predajnoj i prijemnoj strani ista tada za oba važi ograničenje da budu manji ili jednaki od polovine od 2^K (2^{K-1}).

7.4.2 – Selektivna retransmisija



7.4.2 – Selektivna retransmisija

Primer: *Pretpostavimo da **A** šalje 8 okvira numerisanih (0-7) uređaju **B***

- Uzmimo za primer da je K=3 a da je veličina prozora za prijem okvira veći od polovine 2^K tj 5.
- Neka A šalje okvire 0-3 i B ih primi jer mu je veličina prozora 5.
- B šalje potvrdu da je ispravno primio okvire 0-3 i postavlja bafer za prijem okvira 4,5,6,7,0.
- Međutim A ne prima potvrdu i ponovo šalje iste okvire 0-3.
- B sada prihvata te okvire i ne shvata da je A ponovio iste okvire pa prihvata okvir 0 kao početni okvir novog prozora i pravi grešku.

7.4.2 – Selektivna retransmisija

Primer: *Pretpostavimo da **A** šalje 8 okvira numerisanih (0-7) uređaju **B***

- Sličan problem može da se desi ako veličina prijemnog prozora ispunjava ograničenje ali ne i veličina prozora na strani pošiljaoca.
- Neka je prozor kod A veličine 5 a prozor B je 4.
- A sada šalje okvire od 0-4.
- B može da prihvati samo okvire 0-3.
- Ako predpostavimo da okvir 4 kasni.
- Kada okviri 0-3 stignu do B on pomera svoj prozor da bi mogao da prihvati okvire 4-7.
- Sada stiže okvir 4 koji se prihvata u novom prozoru u B.
- Prozor se ponovo pomera u B da bi prihvatio okvire 5,6,7,0.
- Kako potvrda prijema ne stigne do A on ponovo šalje okvire 0-4 i dolazi do greške.

7.4.3 – Efikasnost protokola

- Kompletna analiza protokola klizajućih prozora je mnogo teža nego kod prva dva protokola jer mnogi faktori utiču na efektivnu brzinu prenosa.
- To su pre svega učestalost gubljenja, ošećenje okvira, vrednost tajmera za utvrđivanje kada se šalje ACK i broj okvira sa podacima koji putuju sa uključenom porukom.
- Kada razgovaramo o efikasnosti ovog protokola moramo to da razmatramo pod predpostavkom da nema izgubljenih ili oštećenih okvira
- Da se maksimalno koriste *piggyback* potvrde tj. da nema ACK poruka
- Analiza pokazuje da se pod istim uslovima efektivna brzina prenosa podataka kod klizajućih prozora nalazi negde između brzine za **neograničen** i ***stop and wait*** protokol.

Hvala na pažnji !!!

Pitanja

? ? ?